Amendments to the Specification:

Please replace the title with the following amended title:

METHOD FOR PRODUCING A [[VIABLE]] SPEECH RENDITION OF TEXT FROM DIPHONE SOUNDS

Please replace paragraph [0016] (page 4, line 19 through page 6, line 14) with the following amended paragraph:

[0016] In Phase 1 of our project, we developed: a parser program in Qbasic; a file of over 10,000 individually recorded common words; and a macro program to link a scanning and optical character recognition program to these and a wav player so as to either say or spell each word in the text. We tested many different articles by placing them into the scanner and running the program. We found that of the 20 articles we placed into the scanner, 86% of the words were recognized by our program from the 10,000 word list. Our major focus for Phase 2 of our project has been on the goal of increasing accuracy. Our 86% accuracy with phase one was reasonable, but this still meant that, on average, one to two words per line had to be spelled out, which could interrupt the flow of the reading and make understanding the full meaning of a sentence more difficult. We have found some dictionaries of words of the English language with up to 250,000 words. To record all of them would take over 1,000 hours and still would not cover names, places, nonsense words or expressions like "sheesh", slang like "jumpin", or new words that are constantly creeping into our language. If we recorded a more feasible 20,000 new words, it would probably only have increased the accuracy by 1 to 2%. A new approach was needed. We felt the most likely approach to make a more dramatic increase would involve phonetics. Any American English word can be reasonably reproduced as some combination of 39 phonetic sounds (phonemes). We researched phonetics and experimented linking together different phonemes, trying to create understandable words with them. Unfortunately, the sounds did not sound close enough to the appropriate word, rendering the process infeasible. Most spoken words have a slurring transition from one phoneme to the next. When this transition is missing, the sounds are disjointed and the word is not easily recognized. Overlapping phoneme

wav files by 20% helped, [[a]] but [[,]] not enough. Other possibilities then considered were the use of syllables or groupings of 2 or 3 phonemes (diphones and triphones). Concatenations of all of these produced a reasonable approximation of the desired word. The decision to use diphones was based on practicality. Only diphones are needed as opposed to 100,000 triphones. Due to the numbers, we elected to proceed with using diphones. The number could be further reduced by avoiding combinations that never occur in real words. We elected to include all combinations because names, places and nonsense words can have strange combinations of sounds and would otherwise need to be spelled out. By experimentation, we found that simply saying the sound did not work well. This produced too many accentuated sounds that did not blend well. What worked best was cutting the diphone from the middle of a word, using a good ear and a wav editor to cut the sound out of the word. We initially tried to cut the diphones from 12 or more letter words, since long words would potentially have more diphones in them, but there was so much duplication that we shortly switched to a more methodical method of searching a phonetic dictionary for words with a specific diphone, cutting out that single diphone, and then going on to the next one on the list. If no word could be found, we would create a nonsense word with the desired diphone in the middle of the word, and then extract it with the editor. A considerable amount of time was spent perfecting the process of extracting the diphones. We needed to get the tempo, pitch, and loudness of each recording as similar as possible to the others in order to allow good blending.

Please replace paragraph [0018] (page 6, line 20 through page 7, line 9) with the following amended paragraph:

[0018] We next needed an algorithm to determine which phonemes and diphones to give for a given word. We first explored English texts and chapters dealing with pronunciation rules. Though many rules were found, they were not all inclusive and had many exceptions. We next searched the Internet for pronunciation rules and found an article by the Naval Research Laboratory "Document AD/A021 929 published by National Technical Information Services". It would have required hundreds of nested if-then statements and reportedly it still had only mediocre performance. We decided to try to create our own set of pronunciation rules by working backwards from a phonetic dictionary. We were able to find ~~one online~~ such a

dictionary (CMU dictionary (v0.6)) at the website identified by the uniform resource locator (URL) "ftp://ftp.cs.cmu.edu/project/speech/dict." It had over 100,000 words followed by their phonetic representations ~~representation~~. The site made it clear this was being made freely available for anyone's use. ~~"CMU dictionary (v0.6) ftp://ftp.cs.cmu.edu/project/speech/dict".~~

Please replace paragraph [0021] (page 8, line 21 through page 9, line 10) with the following amended paragraph:

[0021] We next turned our attention to solving other problems that can plague a text to speech program. Homographs are words that have the same spelling but different pronunciation depending on context. For the word "wind" it is necessary to know whether it should be pronounced ~~pronounce~~ like "blowing in the wind" or like "to wind the clock". The most common type homographs have one word as a noun and the other as a verb. We decided to use part of speech context to determine which was more likely in a given sentence. Searching the Internet, we found a public domain dictionary of 230,000 words with their parts of speech. The dictionary is entitled the "Moby Part-of-Speech Dictionary" and is located at the website identified by the URL "ftp://ftp.dcs.shef.ac.uk/share/ilash/Moby/mpos.tar.Z." ~~"Moby Part-of-Speech Dictionary at ftp://ftp.dcs.shef.ac.uk/share/ilash/Moby/mpos.tar.Z".~~ We used this to create a decision matrix that looks at the part of speech of two words before and two after the given word to give the most likely part of speech for the given word. We primed this decision matrix by analyzing sentences from several novels. This result yielded almost a 70% likelihood of getting the right pronunciation.

Please replace paragraph [0028] (page 10, line 4) with the following amended paragraph:

[0028] 3) Convert the 46 phonemes ~~phoneme~~ to numbers.

Please replace paragraph [0034] (page 10, lines 12-13) with the following amended paragraph:

[0034] 8) Find words ~~works~~ that contain a given diphone so we can use that word to create the diphone database.

Please replace paragraph [0044] (page 11, lines 8-9) with the following amended paragraph:

[0044] 18) Macro programmed at the click of one button for [[to]] scanner to scan a document, run OCR, turn it into text, direct the text to be saved in a file, and run our [[or]] Visual Basic program.

Please replace paragraph [0045] (page 11, lines 10-11) with the following amended paragraph:

[0045] 19) Macro programmed to select the text or email currently on the screen, to direct the text to be saved in a file, and run our [[or]] Visual Basic program.

Please replace paragraph [0049] (page 12, lines 8-15) with the following amended paragraph:

[0049] Referring now to the drawing, Figure 1 is a flow diagram of the algorithm used to produce a viable speech rendition of text. The flow diagram should be read in conjunction with the source code, which is set forth below. The basic program begins (step 1) with an initialization routine. This initialization routine involves loading a file which contains the phoneme decision matrices and loading a wav (i.e. sound) file containing a list of pre-recorded words. The matrices are used in the operation of the program to decide the appropriate sound for a given letter. Certain other variables suited for use in the program execution which will be apparent to one of skill in the art may also be initialized.

Please replace paragraph [0050] (page 12, line 16 through page 13, line 2) with the following amended paragraph:

[0050] Following initialization, the program loads (step 2) the first sentence from the text file. The sentence is parsed (step 2), or broken up, into the sequence of words which form the sentence. Each word is examined in turn (step 3) according to the criteria in steps 4, 7, and 8. The program uses both whole words (from an exemplary list of, for example, 10,000 words) and also concatenated words (from the linking of diphones). Any word found on the main list is

produced using the sound recording of that entire word. All other words are produced by the concatenation of diphones unless it included a combination of letters and numbers (like "B42") in which case it is spelled out.

Please replace paragraph [0051] (page 13, line 3 through page 14, line 14) with the following amended paragraph:

[0051] The word is first checked against a list of homographs (step 4). If the word is a homograph, the parts of speech of the adjacent words are determined (step 5). Based on a decision tree, the most appropriate sound file is used (step 6). Alternatively, the word is checked against a list of pre-recorded words (step 7). If the word is contained in the list, the appropriate sound file is used (step 6). If the word is not on either list, the word is checked to see if it contains a combination of numbers and letters (step 8). If so, it is spelled out (step 9). Otherwise, the phoneme rules and a diphone database are used to create the sound file for the word (step 10). The phoneme rules create an algorithm to determine which phonemes and diphones to use for a given word based on pronunciation rules. A set of pronunciation rules was created by working backwards from the CMU phonetic dictionary found on the Internet containing over 100,000 words followed by their phonetic representation. The letter to phoneme rules database was created from the phonetic representations of the words from this phonetic dictionary. The representations are used as data for the letter to phoneme rules which use the phoneme decision matrices. The diphone database consists of combinations of two of the 46 phonemes, making a total of 46 x 46 files. The pronunciation rules are incorporated in the diphone concatenation. Prosodrome variation and homograph discrimination are also used to correctly pronounce words and sentences. Our strategy was to have every letter of each word represented by a single phoneme, and then to find the most common phoneme representation of a letter when one knew certain letters that preceded and followed it. Not all words have the same number of letters as phonemes, so we first had to go through the list and insert blank phonemes when there were too many letters for the original number of phonemes (e.g. for "ph", "th", "gh" or double consonants, the first letter carries the phoneme of the sound made and the second letter would be the blank phoneme). In other cases we combined two phonemes into one in the less common situation when there were too many phonemes for the number of letters in the word.

These manipulations left us with a dictionary of words and matching phonemes; each letter of each word now had a matching phoneme. We used this aligned dictionary as input for a Visual Basic program which determined which was the most common phoneme representation for a given letter, taking into account the one letter before and two letters after it. This was stored in a 26x 26x26x26 matrix form and output to a file so it could be input and used in the next program. Our next program tested the effectiveness of this matrix in predicting the pronunciation of each word on the original phoneme dictionary list. This program utilized the letter to phoneme rules of the matrix for each word and then directly compared this with the original phoneme assigned to that letter by the dictionary. It found 52% of the words were given the entirely correct pronunciation, 65% were either totally correct or had just one letter pronounced incorrectly, and overall 90% of all letters were assigned the correct pronunciation.

Please replace paragraph [0054] (page 15, lines 7-9) with the following amended paragraph:

[0054] If the word is the last one in the sentence (step 11), a modified version of the word is used to provide the proper inflection in accordance with the punctuation (step 12). This process is continued until the entire text file is read (step 13).

Please replace the abstract with the following amended abstract (a replacement sheet is also enclosed herewith):

A text-to-speech system utilizes a method for producing a ~~viable~~ speech rendition of text based on dividing some or all [[the]] words of a [[the]] sentence into component diphones. ~~Analysis of the structure of the sentence is utilized to correctly pronounce homographs. Proper inflection, based on the punctuation of the sentence, is added to produce a more appealing rendition. The invention has particular application for permitting blind persons to receive information previously in text form and, hence, unavailable. The invention is highly versatile, user friendly and the overall flow of sounds highly understandable.~~ A phonetic dictionary is aligned so that each letter within each word has a single corresponding phoneme. The aligned dictionary is analyzed to determine the most common phoneme representation of the letter in the context of a string of letters before and after it. The results for each letter are stored in a phoneme rule

matrix. A diphone database is ~~then~~ created [[by]] using a wav ~~waive~~ editor to cut 2,000 distinct diphones out of specially selected words. [[An]] A computer algorithm ~~used with the invention~~ selects a phoneme for each letter. Then, two phonemes ~~at a time~~ are used to create a diphone. [[, which]] Words are then read aloud by concatenating sounds from the diphone database. In one embodiment, diphones are used only when a word is not one of a list of pre-recorded words. ~~Another algorithm is used to distinguish and use the correct form of homograph. Finally, the invention uses prosodromic variation by adding inflection to the final word of a sentence in accordance with punctuation, e.g. question marks, periods and exclamation marks, to enhance the realisticness of the program. Three macros were created to allow aspects of the program to run at the touch of a single button. A 98% accuracy rate has been achieved.~~